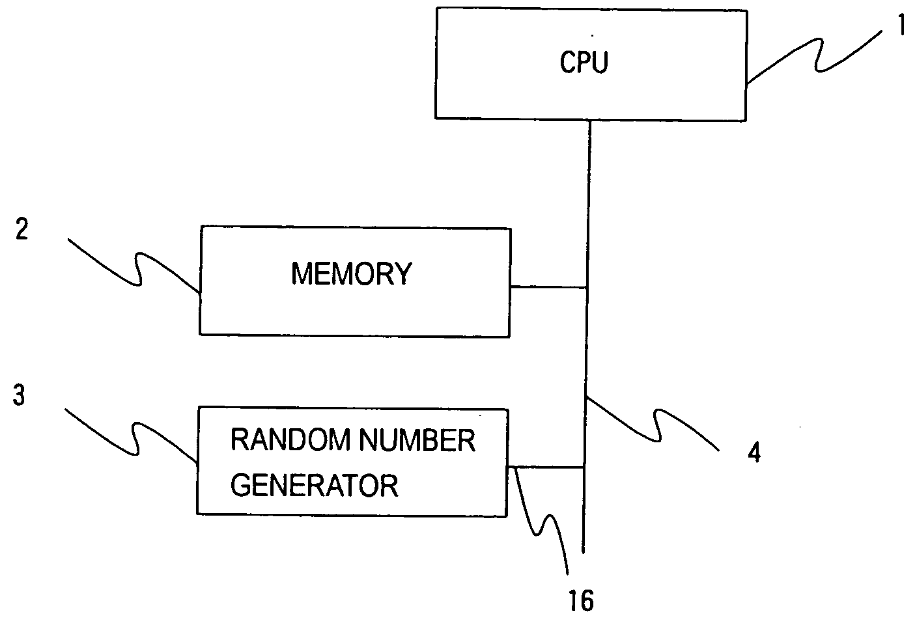


Fig. 1

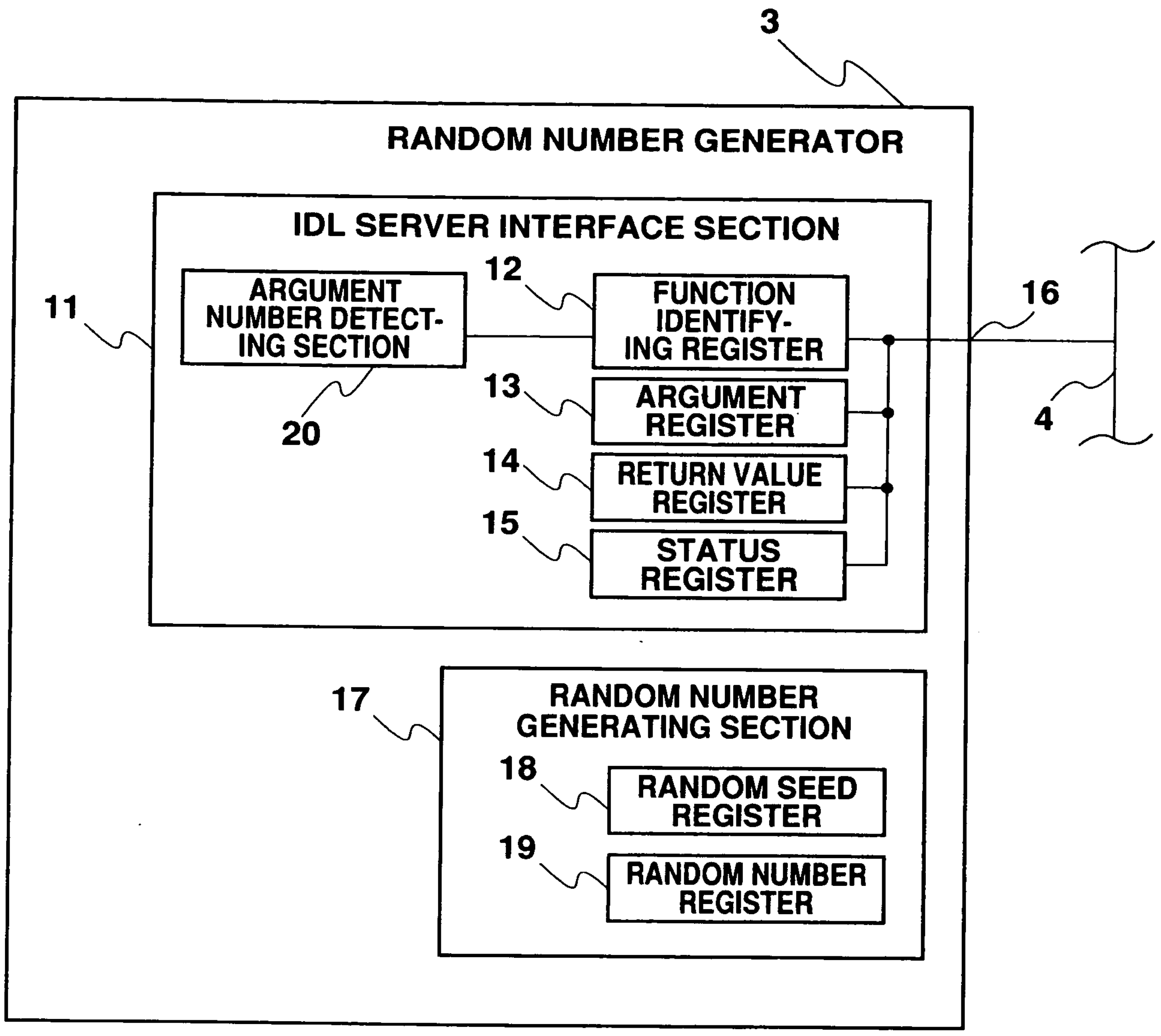

3
RANDOM NUMBER GENERATOR
IDL SERVER INTERFACE SECTION
11
ARGUMENT NUMBER DETECTING SECTION
20
12
FUNCTION IDENTIFYING REGISTER
13
ARGUMENT REGISTER
14
RETURN VALUE REGISTER
15
STATUS REGISTER
16
4
17
RANDOM NUMBER GENERATING SECTION
18
RANDOM SEED REGISTER
19
RANDOM NUMBER REGISTER

Fig. 2

```
interface randomGenerator{
    void  setSeed( in double seed  );
    double getRandom();
};
```

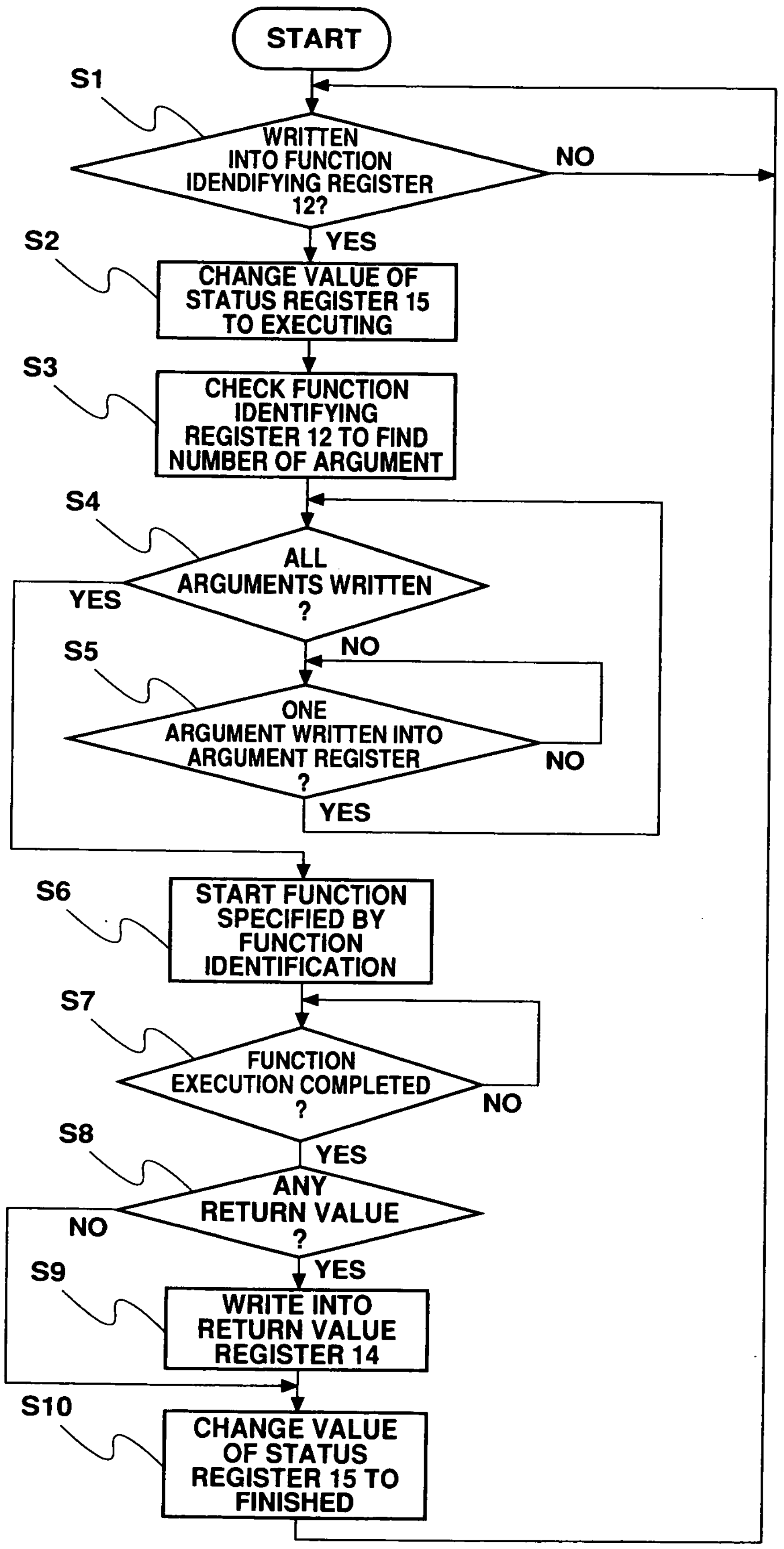
Fig. 3

```
/* part common to IDL interface */
/* status register value */
#define   Executing     1
#define   Finished      2
#define   Requesting    3

/* function for accessing register */
void     putFunctionID( FID );
void     putDoubleArg( double );
double   getDoubleReturn();
int      getStatus();

/* randomGenerator unique part */
/* definition of function identification value */
#define   FID_setSeed        1
#define   FID_getRandom    2

/* proto-type declaration in C-language function, corresponding to function */
void     setSeed( double seed );
double   getRandom()
```

Fig. 4

```
void  setSeed( double seed )
{
    putFunctionID( FID__setSeed );
    putDoubleArg( seed );
}

double  getRandom( )
{
    putFunctionID( FID__getRandom );
    while( getStatus() != Finished )
        ;
    return getDoubleReturn( );
}
```

Fig. 5


START
S1
WRITTEN INTO FUNCTION IDENDIFYING REGISTER 12?
NO
YES
S2
CHANGE VALUE OF STATUS REGISTER 15 TO EXECUTING
S3
CHECK FUNCTION IDENTIFYING REGISTER 12 TO FIND NUMBER OF ARGUMENT
S4
ALL ARGUMENTS WRITTEN ?
YES
NO
S5
ONE ARGUMENT WRITTEN INTO ARGUMENT REGISTER ?
NO
YES
S6
START FUNCTION SPECIFIED BY FUNCTION IDENTIFICATION
S7
FUNCTION EXECUTION COMPLETED ?
NO
YES
S8
ANY RETURN VALUE ?
NO
YES
S9
WRITE INTO RETURN VALUE REGISTER 14
S10
CHANGE VALUE OF STATUS REGISTER 15 TO FINISHED
Fig. 6

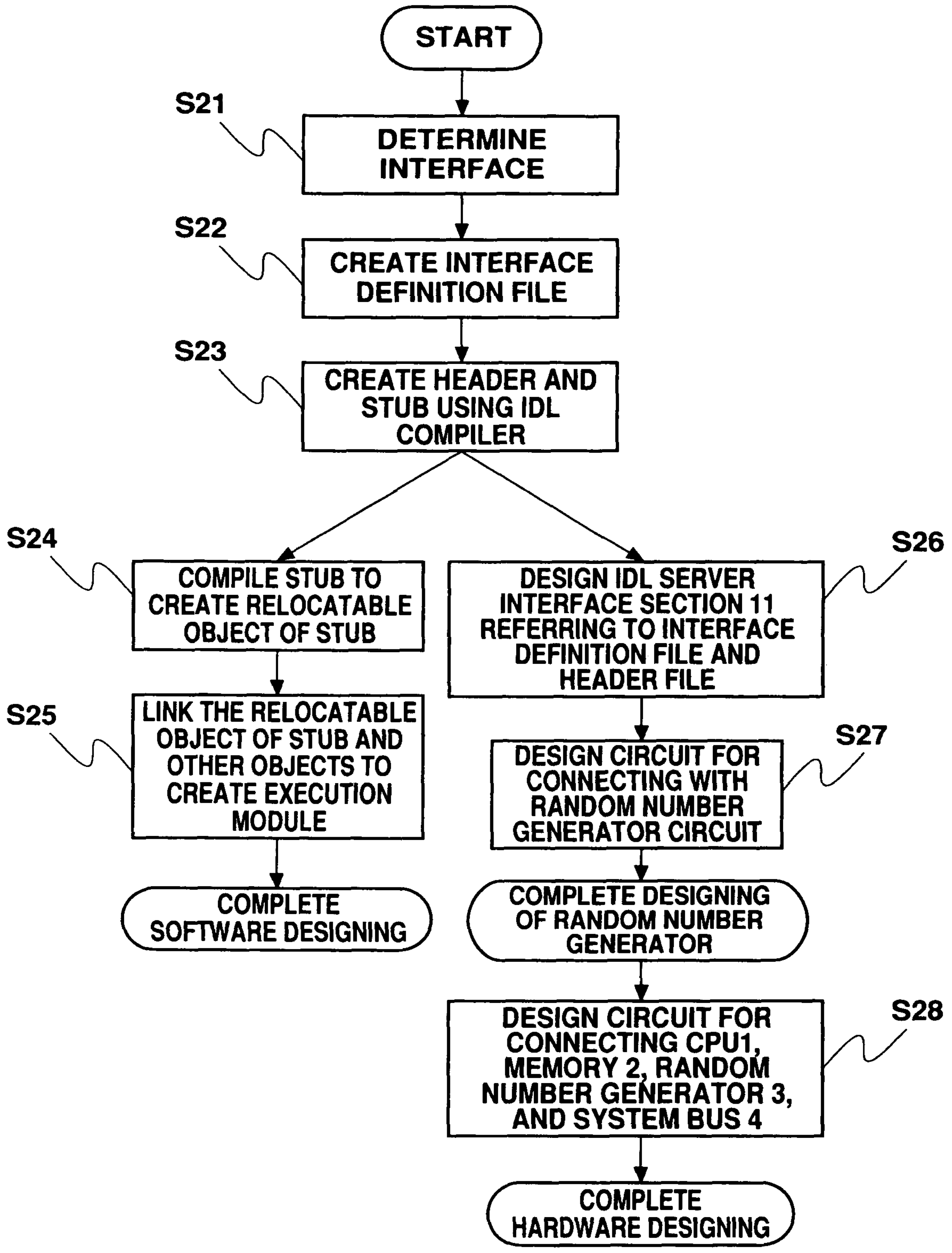

START
S21
DETERMINE INTERFACE
S22
CREATE INTERFACE DEFINITION FILE
S23
CREATE HEADER AND STUB USING IDL COMPILER
S24
COMPILE STUB TO CREATE RELOCATABLE OBJECT OF STUB
S25
LINK THE RELOCATABLE OBJECT OF STUB AND OTHER OBJECTS TO CREATE EXECUTION MODULE
COMPLETE SOFTWARE DESIGNING
S26
DESIGN IDL SERVER INTERFACE SECTION 11 REFERRING TO INTERFACE DEFINITION FILE AND HEADER FILE
S27
DESIGN CIRCUIT FOR CONNECTING WITH RANDOM NUMBER GENERATOR CIRCUIT
COMPLETE DESIGNING OF RANDOM NUMBER GENERATOR
S28
DESIGN CIRCUIT FOR CONNECTING CPU1, MEMORY 2, RANDOM NUMBER GENERATOR 3, AND SYSTEM BUS 4
COMPLETE HARDWARE DESIGNING

Fig. 7

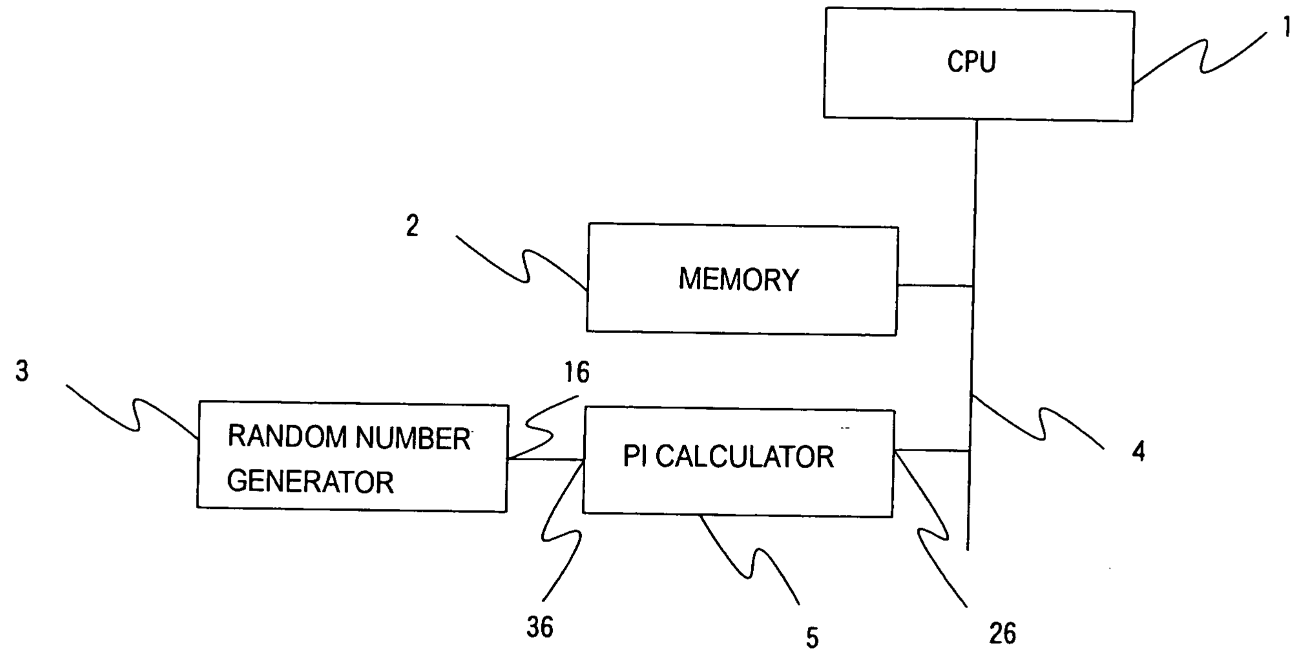


Fig. 8

```
interface piCalculator{
    void  setCount( in int count );
    double getPi();
};
```

Fig. 9

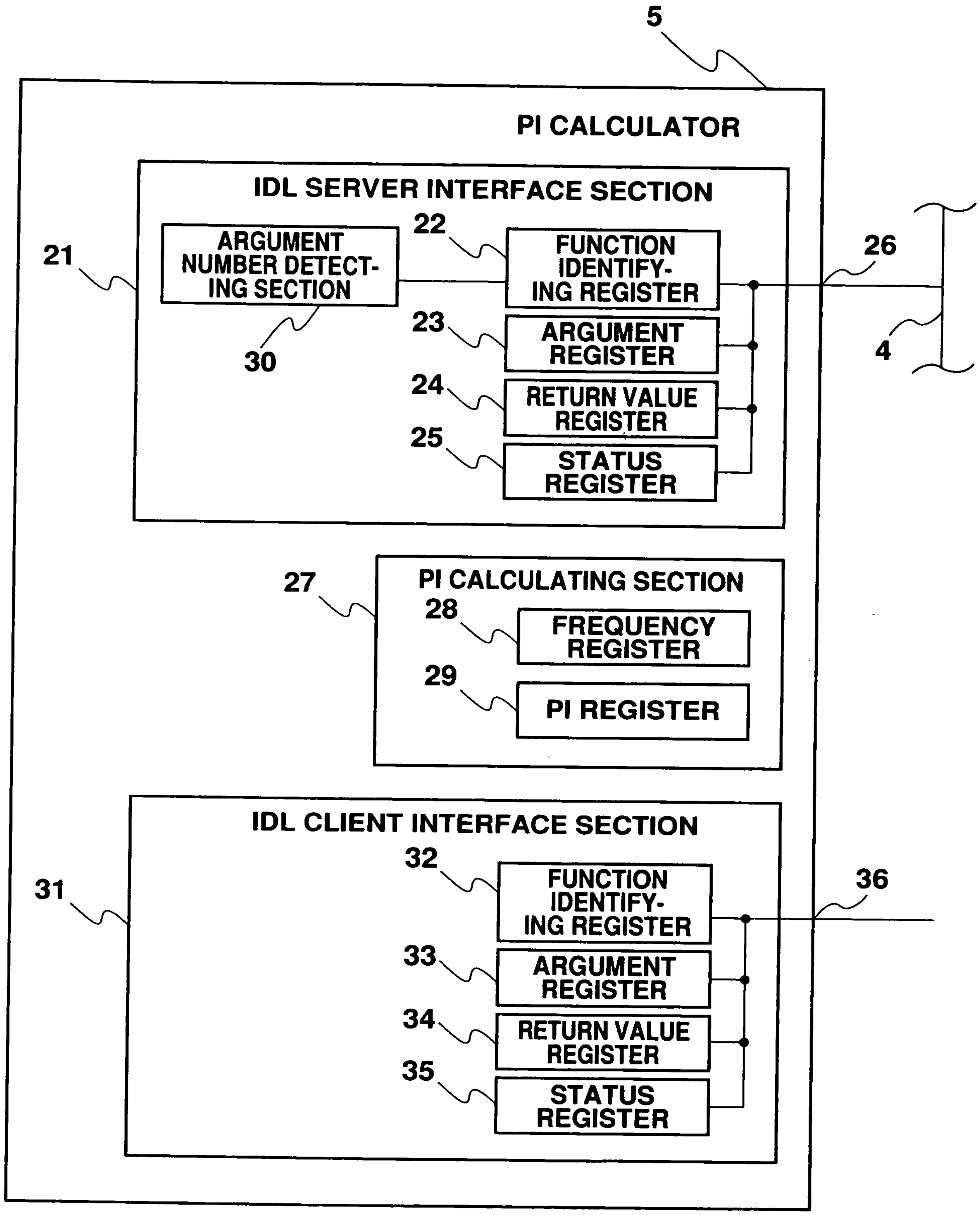

5
PI CALCULATOR
IDL SERVER INTERFACE SECTION
21
ARGUMENT NUMBER DETECT-ING SECTION
30
22
FUNCTION IDENTIFY-ING REGISTER
23
ARGUMENT REGISTER
24
RETURN VALUE REGISTER
25
STATUS REGISTER
26
4
27
PI CALCULATING SECTION
28
FREQUENCY REGISTER
29
PI REGISTER
IDL CLIENT INTERFACE SECTION
31
32
FUNCTION IDENTIFY-ING REGISTER
33
ARGUMENT REGISTER
34
RETURN VALUE REGISTER
35
STATUS REGISTER
36

Fig. 10

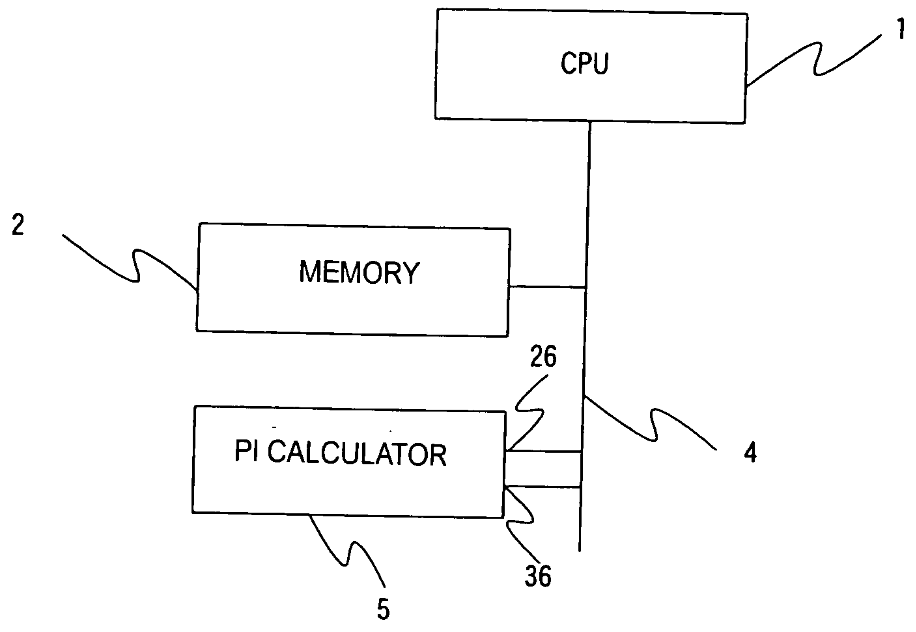


Fig. 11

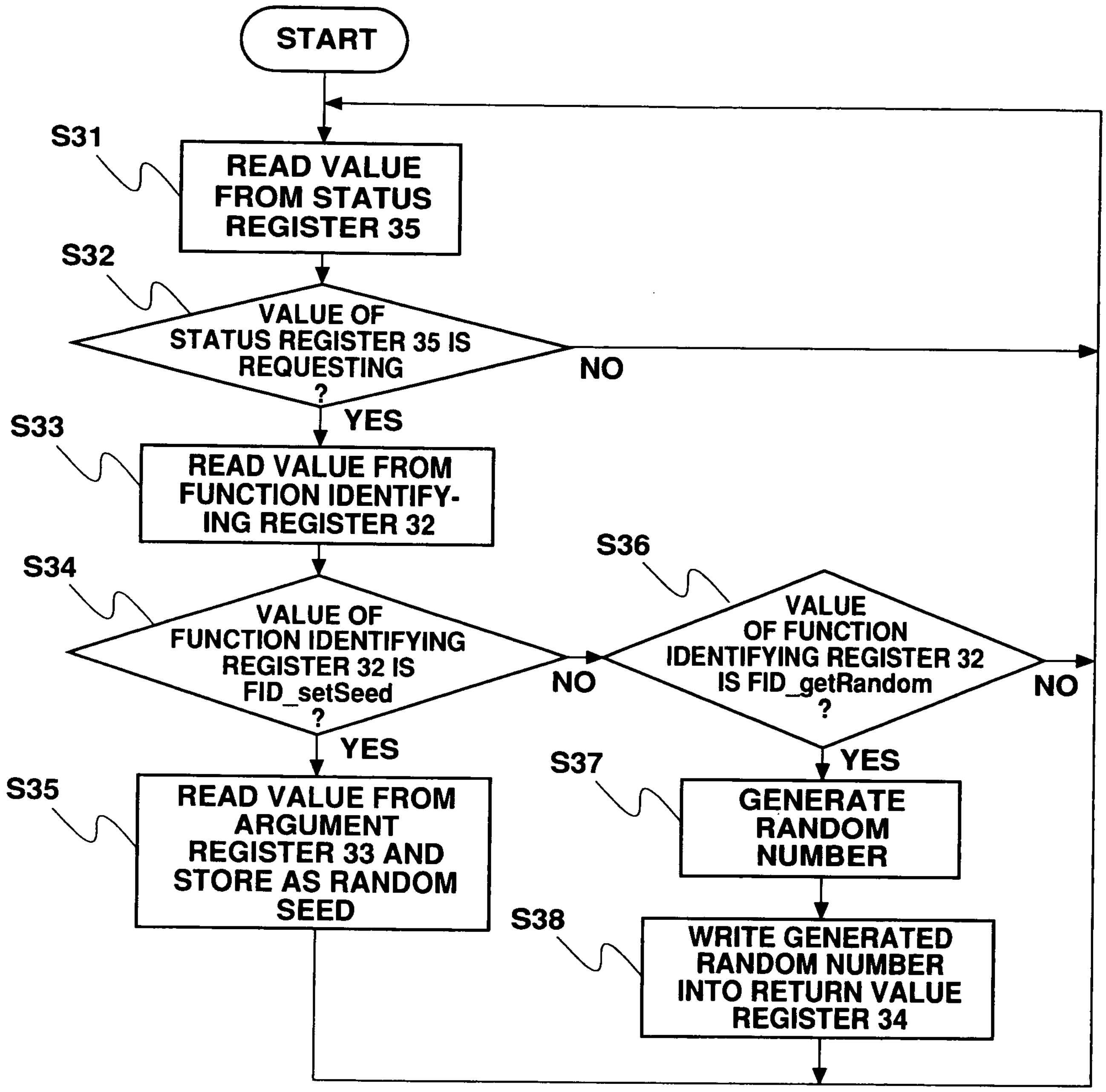

START
S31
READ VALUE FROM STATUS REGISTER 35
S32
VALUE OF STATUS REGISTER 35 IS REQUESTING ?
NO
YES
S33
READ VALUE FROM FUNCTION IDENTIFY-ING REGISTER 32
S34
VALUE OF FUNCTION IDENTIFYING REGISTER 32 IS FID_setSeed ?
NO
YES
S35
READ VALUE FROM ARGUMENT REGISTER 33 AND STORE AS RANDOM SEED
S36
VALUE OF FUNCTION IDENTIFYING REGISTER 32 IS FID_getRandom ?
NO
YES
S37
GENERATE RANDOM NUMBER
S38
WRITE GENERATED RANDOM NUMBER INTO RETURN VALUE REGISTER 34

Fig. 12

```
/* part common to IDL interface */
/* status register value */
#define   Waiting      0
#define   Executing    1
#define   Finished     2
#define   Requesting   3

/* function for accessing register */
FID       getFunctionID( );
double    getDoubleArg();
double    putDoubleReturn( double );
int       getStatus();

/* randomGenerator unique part */
/* definition of function identification value */
#define   FID_setSeed      1
#define   FID_getRandom    2

/* proto-type declaration in C-language function, corresponding to function */
void      setSeed( double seed );
double    getRandom()
```

Fig. 13

```
main()
{
    while(1){
        while( getStatus() != Requesting )
            ;
        switch( getFunctionID() ){
            case FID_setSeed:
                setSeed( getDoubleArg() );
                break;
            case FID_getRandom:
                putDoubleReturn( getRandom() );
                break;
        }
    }
}
```

Fig. 14

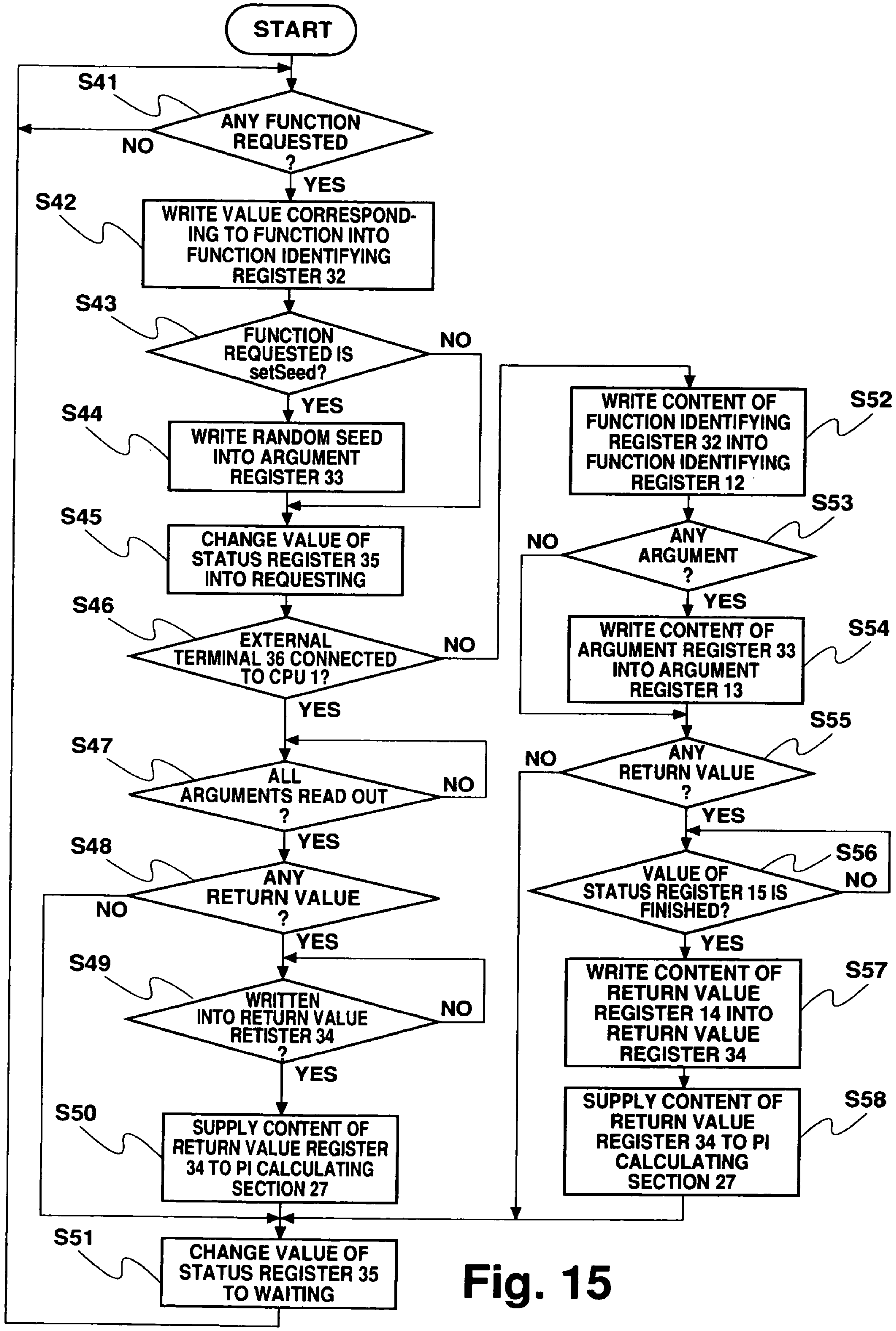

START
S41
ANY FUNCTION REQUESTED ?
NO
YES
S42
WRITE VALUE CORRESPONDING TO FUNCTION INTO FUNCTION IDENTIFYING REGISTER 32
S43
FUNCTION REQUESTED IS setSeed?
NO
YES
S44
WRITE RANDOM SEED INTO ARGUMENT REGISTER 33
S45
CHANGE VALUE OF STATUS REGISTER 35 INTO REQUESTING
S46
EXTERNAL TERMINAL 36 CONNECTED TO CPU 1?
NO
YES
S47
ALL ARGUMENTS READ OUT ?
NO
YES
S48
ANY RETURN VALUE ?
NO
YES
S49
WRITTEN INTO RETURN VALUE RETISTER 34 ?
NO
YES
S50
SUPPLY CONTENT OF RETURN VALUE REGISTER 34 TO PI CALCULATING SECTION 27
S51
CHANGE VALUE OF STATUS REGISTER 35 TO WAITING
S52
WRITE CONTENT OF FUNCTION IDENTIFYING REGISTER 32 INTO FUNCTION IDENTIFYING REGISTER 12
S53
ANY ARGUMENT ?
NO
YES
S54
WRITE CONTENT OF ARGUMENT REGISTER 33 INTO ARGUMENT REGISTER 13
S55
ANY RETURN VALUE ?
NO
YES
S56
VALUE OF STATUS REGISTER 15 IS FINISHED?
NO
YES
S57
WRITE CONTENT OF RETURN VALUE REGISTER 14 INTO RETURN VALUE REGISTER 34
S58
SUPPLY CONTENT OF RETURN VALUE REGISTER 34 TO PI CALCULATING SECTION 27

Fig. 15